Problem 9 (20 points)

Problem Description

In this problem you will use `sklearn.svm.SVR` to train a support vector machine for a regression problem. Your model will predict G forces experienced by a sports car as it travels through a chicane in the Nurburgring.

Fill out the notebook as instructed, making the requested plots and printing necessary values.

You are welcome to use any of the code provided in the lecture activities.

Summary of deliverables:

Results:

- Plot the fitted SVR function for three different epsilon values
- Compute the R2 score for each of the fitted functions

Discussion:

- Discuss the performance of the models and the effect of epsilon

Imports and Utility Functions:

In [5]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVR

def plot_data(X, y, ax = None):
    if ax is None:
        ax = plt.gca()
        showflag = True
    else:
        showflag = False
    ax.scatter(X,y, c = 'blue')
    ax.set_xlabel('Normalized Position')
    ax.set_ylabel('G Force')
    if showflag:
        plt.show()
    else:
        return ax

def plot_svr(model, X, y):
    ax = plt.gca()
    ax = plot_data(X, y, ax)
    xs = np.linspace(min(X), max(X), 1000).reshape(-1,1)
    ys = model.predict(xs)
    ax.plot(xs,ys,'r-')
    plt.legend(['Data', 'Fitted Function'])
    plt.show()
```

Load and visualize the data

The data is contained in `nurburgring.npy` and can be loaded with `np.load()`. The first column corresponds to the normalized position of the car in the chicane, and the second column corresponds to the measured G force experienced at that point in the chicane.

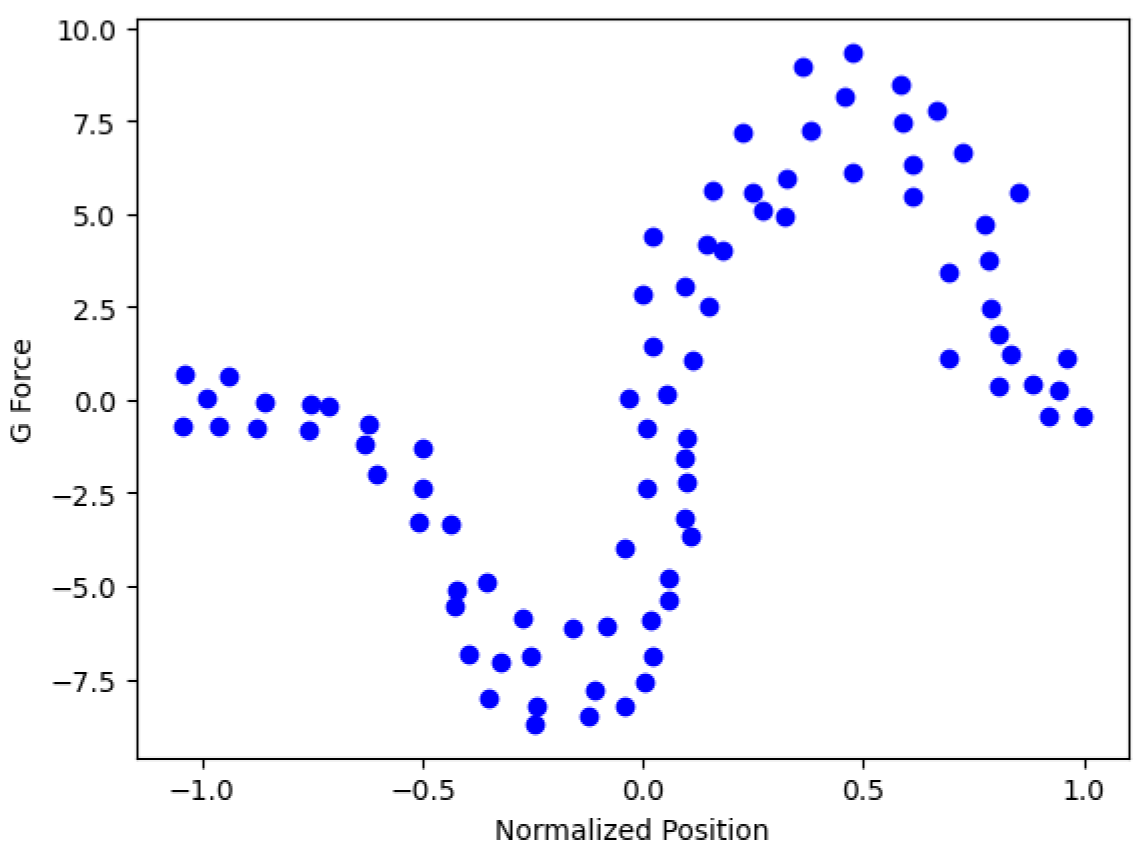
Store the data as:

- `X` (Nx1) array of position data
- `y` N-dimensional vector of G force data

Then visualize the data with `plot_data(X,y)`

Note: use X.reshape(-1,1) to make the `X` array two dimensional as required by 'SVR.fit(X,y)'

In [6]:
```
# YOUR CODE GOES HERE

nurburgring = np.load("data/nurburgring.npy")

X = nurburgring[:,0].reshape(-1,1)
y = nurburgring[:,1]

plot_data(X,y)
```

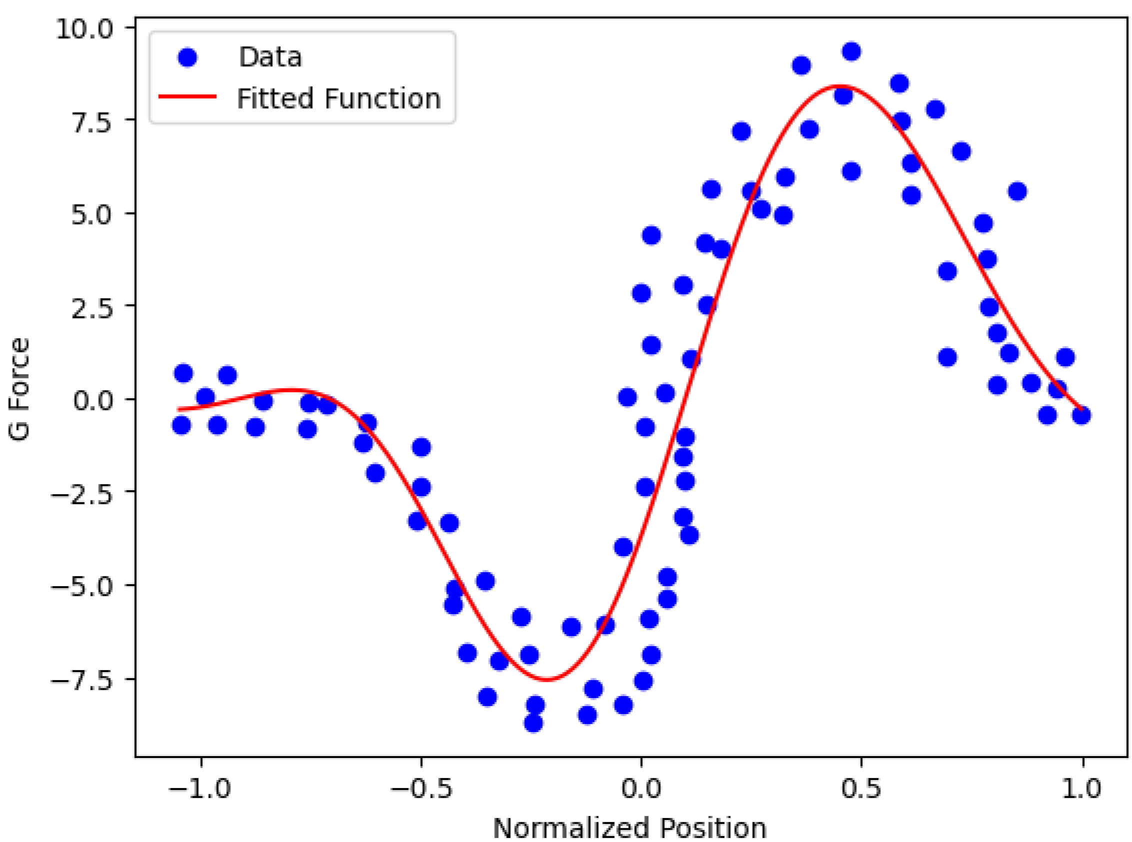


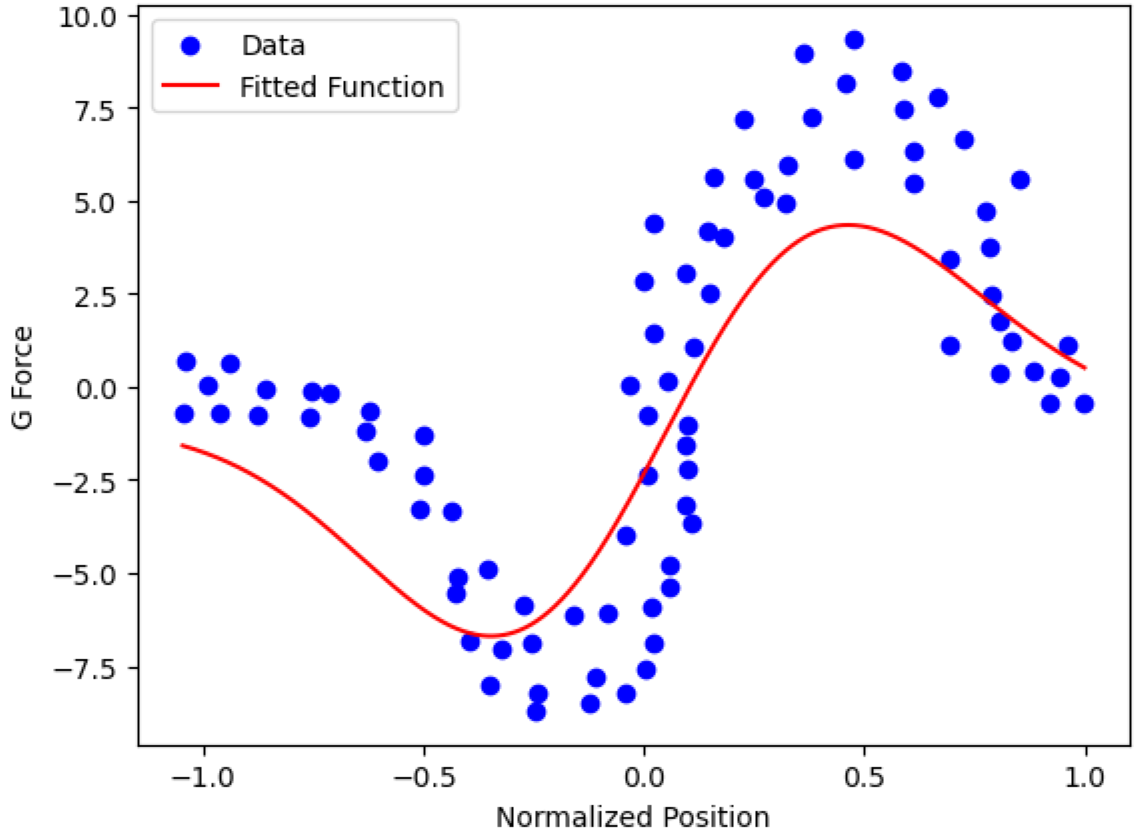
Train Support Vector Regressors

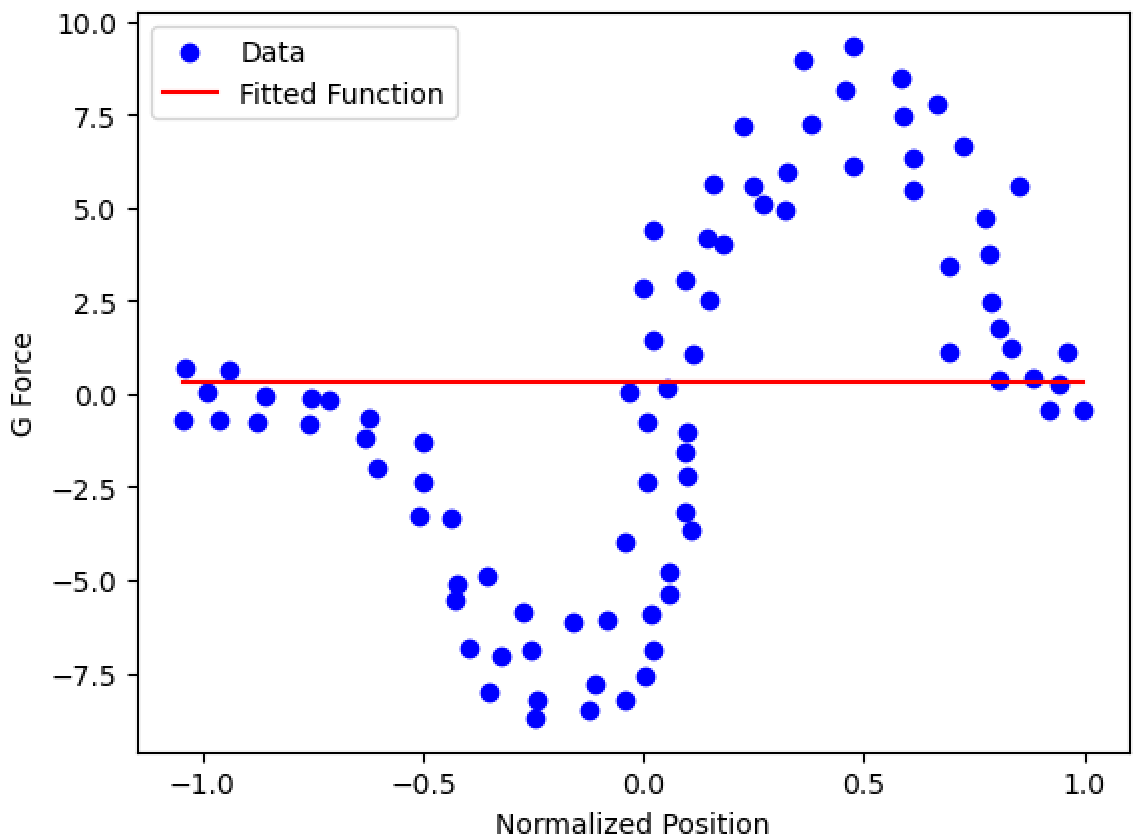
Train three different support vector regressors using the RBF Kernel, C = 100, and epsilon = [1, 5, 10]. For each model, report the coefficient of determination (R^2) for the fitted model using the builtin sklearn function} `model.score(X,y)`, and plot the fitted function against the data using `plot_svr(model, X, y)`

In [8]:
```
# YOUR CODE GOES HERE
e = [1,5,10]
models = []
R2 = []
for i in range(len(e)):
    model = SVR(kernel="rbf",C=100,epsilon=e[i])
    model.fit(X,y)
    models.append(model)
    R2.append(model.score(X,y))

k = 0
for i in R2:
    print(f'for e = {e[k]}, R2 is', i,"\n")
    k+=1

for i in models:
    plot_svr(i,X,y)
```

```
for e = 1, R2 is 0.803896252951426

for e = 5, R2 is 0.6412302705136446

for e = 10, R2 is -0.005585141758953638
```




Data
Fitted Function
G Force
Normalized Position
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−1.0
−0.5
0.0
0.5
1.0


Data
Fitted Function
G Force
Normalized Position
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−1.0
−0.5
0.0
0.5
1.0

Discussion

Briefly discuss the performance of the three models, and explain how the value of epsilon influences the fitted model within the context of epsilon insenstive loss introduced in lecture.

The epsilon value affects the model fit of the data. The smaller epsilon leads to a hard constrain which can led to a good fit of the data. It leads to less errors as it reduced the generalizaton of the data. As the epsilon increases, the model's performance decreases and it can leads to errors which therefore makes the data too generalized and therefore not giving a smooth fit to the data.

In [ ]: